Regression Computation Assignment

The assignment is due on Thursday June 4 at 23:59.

Purpose and Overview

The purpose of this assignment is for you to demonstrate your understanding of the mathematics and computational procedures involved in the estimation of basic regression models. Below are several regression models. The first two (`m1` and `m2`) are OLS models explaining life expectancy and the latter two (`m3` and `m4`) are logistic regression models explaining democracy. All data are drawn from the Quality of Government Basic Dataset. The data and codebook are available here: `http://qog.pol.gu.se/data/datadownloads/qogbasicdata`.

```
d <- rio::import("http://www.qogdata.pol.gu.se/data/qog_bas_cs_jan15.csv")

f1 <- une_leb ~ I(gle_cgdpc/1e4) + I(ross_oil_netexpc/1e3)
m1 <- lm(f1, data = d)

f2 <- une_leb ~ I(gle_cgdpc/1e4)*factor(chga_demo) +
                I(ross_oil_netexpc/1e3) + factor(ht_colonial)
m2 <- lm(f2, data = d)

f3 <- chga_demo ~ al_ethnic
m3 <- glm(f3, data = d, family = binomial(link='logit'))

f4 <- chga_demo ~ al_ethnic + I(log(wdi_landarea)) +
                  I(gle_cgdpc/1e4) + I(ross_oil_netexpc/1e3)
m4 <- glm(f4, data = d, family = binomial(link='logit'))
```

Your task is to reproduce parts of these analyses using R (or Stata), without the use of the `lm`, `glm`, `lm.fit`, or similar ready-made functions (or, for Stata, without the use of `reg`, `logit`, `glm`, or similar commands). The instructions below outline the code you should produce. You may use R (based on techniques from lecture) or Stata (for relevant technical details on Stata, see Ch. 11 and Appendix A from Cameron and Trivedi 2010).

Submitting Your Assignment

Submit your assignment via email to Thomas (`mailto:tleeper@ps.au.dk`) in the form of a single, complete R syntax file (.R) or Stata (.do) file in your submitted assignment. Each step of the code should be numbered and labeled (in the form of a R or Stata comment) according to the numbering of the steps below. You do not need to explain, describe, or present the output of any analyses.

Ordinary Least Squares Regression

1. Construct a design matrix **X** for `m1`. Extract the **R** matrix from the **QR** decomposition of **X**. Write code to produce this matrix using the formula given in class for the inverse of a 3-by-3 matrix.

```
# Design matrix
dat <- cbind(1, d$gle_cgdpc/1e4, d$ross_oil_netexpc/1e3)
cc <- complete.cases(dat)
X <- dat[cc,]

# QR decomposition
QR <- qr(X)
R <- qr.R(QR)
R

# inverse of the R matrix, manually
matrix(c(R[2,2] * R[3,3] - R[2,3] * R[3,2],
         -(R[2,1] * R[3,3] - R[2,3] * R[3,1]),
         R[2,1] * R[3,2] - R[2,2] * R[3,1],
         -(R[1,2] * R[3,3] - R[1,3] * R[3,2]),
         R[1,1] * R[3,3] - R[1,3] * R[3,1],
         -(R[1,1] * R[3,2] - R[1,2] * R[3,1]),
         R[1,2] * R[2,3] - R[1,3] * R[2,2],
         -(R[1,1] * R[2,3] - R[1,3] * R[2,1]),
         R[1,1] * R[2,2] - R[1,2] * R[2,1]), nrow = 3) * (1/det(R))

# check inverse of the R matrix, automatically
solve(R)
```

2. Create a vector `y` to represent the outcome. Then construct an appropriate design

matrix **X** for `m2` (as represented by formula `f2`) from the original variables in the dataset, including the specified categorical (factor) variables and interaction term (see Fox pp.153–155). Note: Be careful handling missing values!

```
# define `y`
y <- d$une_leb


# call `model.matrix` directly to create design matrix:
X <- model.matrix(f2, data = d)


# manual alternative:
x1 <- d$gle_cgdpc/10000
x2 <- d$chga_demo
x3 <- x1*x2
x4 <- d$ross_oil_netexpc/1000
u <- sort(unique(d$ht_colonial))
x5 <- matrix(NA, nrow = nrow(d), ncol = length(u))
for(i in seq_along(u)) {
    x5[,i] <- as.numeric(d$ht_colonial == u[i])
}
dat <- cbind(y, 1, x1, x2, x3, x4, x5)
cc <- complete.cases(dat)
X <- dat[cc, -1]
colnames(X) <- c('Intercept', 'gle_cgdpc', 'chga_demo',
                 'gle_cgdpc*chga_demo', 'ross_oil_netexpc',
                 paste0('ht_colonial', u))
# drop linearly dependent columns
X <- X[, !colnames(X) %in% c('ht_colonial0', 'ht_colonial9')]


# pass complete cases to `y`
```

```
y <- dat[cc, 1]
```

3. Estimate the Ordinary Least Squares (OLS) coefficients in `m2` using matrix notation (see Fox p.155). You may use a QR decomposition if you so choose.

```
# using standard matrix notation (X'X)^{-1}X'y
beta <- solve(t(X) %*% X) %*% t(X) %*% y
beta


# using a QR decomposition
decomp <- qr(X)
beta <- solve(qr.R(decomp)) %*% t(qr.Q(decomp)) %*% y
beta
```

4. Obtain the fitted values (where $\hat{\mathbf{y}} = \mathbf{X}\hat{\beta}$), from model `m2` with all variables held at their observed values, except:

   - `gle_cgdpc` = 10000
   - `chga_demo` = 1
   - `ross_oil_netexpc` = 1000
   - `ht_colonial` = 0

```
Xtmp <- X
Xtmp[,'gle_cgdpc'] <- 10000
Xtmp %*% beta


Xtmp <- X
Xtmp[,'chga_demo'] <- 1
Xtmp %*% beta
```

```r
Xtmp <- X
Xtmp[,'ross_oil_netexpc'] <- 1000
Xtmp %*% beta


Xtmp <- X
# redefine all `ht_colonial` columns:
Xtmp[,grepl('ht_colonial',colnames(Xtmp))] <- 0
# or manually:
Xtmp[,'ht_colonial1'] <- 0
Xtmp[,'ht_colonial2'] <- 0
Xtmp[,'ht_colonial3'] <- 0
Xtmp[,'ht_colonial4'] <- 0
Xtmp[,'ht_colonial5'] <- 0
Xtmp[,'ht_colonial6'] <- 0
Xtmp[,'ht_colonial7'] <- 0
Xtmp[,'ht_colonial8'] <- 0
#Xtmp[,'ht_colonial9'] <- 0 # dropped previously
Xtmp[,'ht_colonial10'] <- 0
Xtmp %*% beta
```

5. Estimate the variance-covariance matrix of the OLS coefficient estimates for `m2` and extract estimated standard errors from that matrix (see Fox p.158).

```r
fitted <- X %*% beta
e <- fitted - y
df <- nrow(X) - length(beta)
s2 <- (t(e) %*% e) / df


# variance-covariance matrix
V <- as.numeric(s2) * solve(t(X) %*% X)
```

```r
# standard errors
sqrt(diag(V))
```

6. Use the estimated coefficients and standard errors for `m2` to obtain the t-statistics for each coefficient estimate under a null hypothesis $H_0 : \beta_k = 0$ (see Fox 3.3 and 4.4). Calculate the one-tailed and two-tailed p-values for each test statistic using the `pt` function, which returns the value of Student's t cumulative distribution function.

```r
tstats <- beta/sqrt(diag(V))


# determine degrees of freedom for t-test
ndf <- nrow(X) - length(beta)


# one-tailed test (two alternative strategies)
1-pt(abs(tstats), ndf)
pt(-abs(tstats), ndf)


# two-tailed test (two alternative strategies)
2*(1-pt(abs(tstats), ndf))
2*pt(-abs(tstats), ndf)
```

Logistic Regression

7. Represent the log-likelihood function for logistic regression as an R (or Stata) function (see Fox 3.6). Recall the general structure of the log-likelihood for a binary outcome variable is:

$$\ln \mathcal{L}(\beta) = \sum_{i=1}^{n} y_i \ln(\pi_i) + (1 - y_i) \ln(1 - \pi_i)$$

where π_i in the logistic model is given by the inverse link function $\frac{e^{\mathbf{X}_i\beta}}{1 + e^{\mathbf{X}_i\beta}}$.

```
logitll <- function(beta, y, X){
  p <- exp(X %*% beta)/(1+exp(X %*% beta))
  lik <- y * log(p) + (1-y) * log(1-p)
  sum(lik)
}
```

8. Estimate the coefficients for `m3` using two methods of maximum likelihood estimation: (a) a grid search and (b) an optimization algorithm implemented by `optim`. For the grid search, search for possible slope and intercept coefficients in the range between -5 and +5.

```
dat <- cbind(d$chga_demo,1,d$al_ethnic)
cc <- complete.cases(dat)
Xm3 <- dat[cc,2:3]
ym3 <- dat[cc,1]

# grid search (could be more precise)
iseq <- seq(-5, 5, by = 0.025)
sseq <- seq(-5, 5, by = 0.025)
g <- expand.grid(intercept = iseq, slope = sseq)
```

```r
gs <- apply(g, 1, function(z) logitll(beta = z, X = Xm3, y = ym3))
g[which.max(gs), ]


# optimization algorithm
optm3 <- optim(par = rep(0, 2),
               fn = logitll,
               X = Xm3,
               y = ym3,
               control = list(fnscale = -1),
               hessian = TRUE,
               method = "BFGS")
optm3$par
```

9. Estimate the standard errors for coefficient estimates in `m3` using bootstrapping.

```r
# function to estimate model one time
once <- function(X, y) {
    opt <- optim(par = rep(0, 2),
                 fn = logitll,
                 X = X,
                 y = y,
                 control = list(fnscale = -1),
                 hessian = TRUE,
                 method = "BFGS")
    opt$par
}


# bootstrap
n <- 5000
bootmat <- matrix(numeric(), ncol = 2, nrow = n)
```

```r
for(i in 1:n) {
    s <- sample(1:nrow(X), nrow(X), TRUE)
    Xtmp <- Xm3[s,]
    ytmp <- ym3[s]
    tried <- try(once(Xtmp, ytmp))
    if(!inherits(tried, 'try-error'))
        bootmat[i,] <- tried
}
apply(bootmat, 2, sd, na.rm = TRUE)

# jackknife (just for fun)
jackmat <- matrix(numeric(), ncol = 2, nrow = nrow(X))
for(i in 1:nrow(X)) {
    Xtmp <- Xm3[-i,]
    ytmp <- ym3[-i]
    tried <- try(once(Xtmp, ytmp))
    if(!inherits(tried, 'try-error'))
        jackmat[i,] <- tried
}
apply(jackmat, 2, sd, na.rm = TRUE)
```

10. Obtain 95% bootstrap confidence intervals for the `m3` coefficient estimates from the bootstrap distribution of the coefficients.

```r
t(apply(bootmat, 2, quantile, probs = c(0.025,0.975), na.rm = TRUE))
```

11. Estimate the coefficients for `m4` using `optim`, obtain the standard errors as the diagonal of the matrix inverse of the negative hessian matrix, calculate the z-statistic for each coefficient estimate, and determine its two-tailed p-value from `pnorm` (the normal cumulative distribution function).

```r
dat <- cbind(d$chga_demo,1,d$al_ethnic, log(d$wdi_landarea),
             d$gle_cgdpc/1e4, d$ross_oil_netexpc/1e3)
cc <- complete.cases(dat)
Xm4 <- dat[cc,-1]
ym4 <- dat[cc,1]


optm4 <- optim(par = rep(0, 5),
               fn = logitll,
               X = Xm4,
               y = ym4,
               control = list(fnscale = -1),
               hessian = TRUE,
               method = "BFGS")


# coefficient estimates
optm4$par


# standard errors
sqrt(diag(solve(-optm4$hessian)))


# z-statistics
zstats <- optm4$par/sqrt(diag(solve(-optm4$hessian)))
zstats


# two-tailed p-values (two alternative strategies)
2*(1-pnorm(abs(zstats)))
2*pnorm(-abs(zstats))
```

12. Using your estimated coefficients for `m3` and `m4`, compute the *average* predicted

probability of observing the outcome when `al_ethnic` is set to each of the five-number summary quantiles $(0.00, 0.25, 0.50, 0.75, 1.00)$ of the observed distribution of `al_ethnic` and all other covariates are held at their observed values. Your results should be two vectors, one for each model, each containing five numbers.

```r
# function to compute predicted probabilities
pp <- function(dat, coefvec) {
    mean(exp(dat %*% coefvec)/(1+exp(dat %*% coefvec)))
}
qs <- quantile(d$al_ethnic, c(0.00,0.25,0.50,0.75,1.00), na.rm = TRUE)


# m3
Xm3tmp <- Xm3
m3out <- numeric(5)
for(i in seq_along(qs)) {
    Xm3tmp[,2] <- qs[i]
    m3out[i] <- pp(dat = Xm3tmp, coefvec = optm3$par)
}
m3out


# m4
Xm4tmp <- Xm4
m4out <- numeric(5)
for(i in seq_along(qs)) {
    Xm4tmp[,2] <- qs[i]
    m4out[i] <- pp(dat = Xm4tmp, coefvec = optm4$par)
}
m4out
```